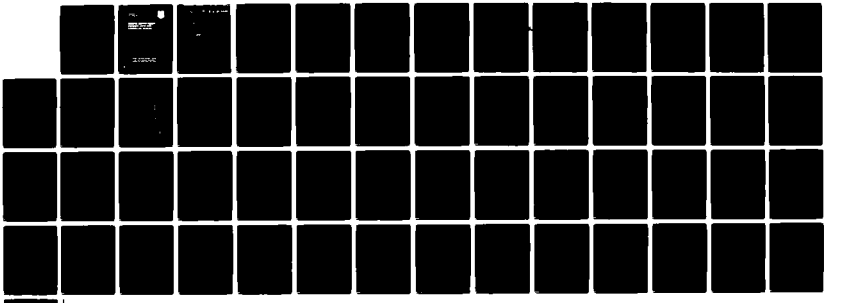
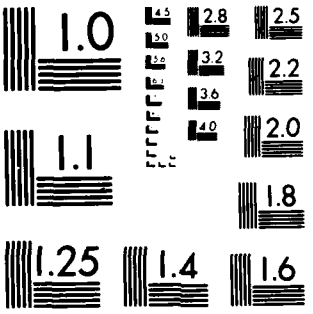
AD-A125 627 INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY 1/1
VOLUME 2 SYSTEM/SUB..(U) GENERAL DYNAMICS FORT WORTH TX
FORT WORTH DIV H C CONN ET AL. JAN 83 DMA-2-014-VOL-2
UNCLASSIFIED RADC-TR-83-3-VOL-2 F30602-81-C-0039 F/G 9/2 NL



END
DATE FILMED
4-83
DTIC

M-2



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

AD A125027

INTERACTIVE COMPUTER ANALYSIS AND DESIGN SYSTEM STUDY
System/Subsystem Specification

General Dynamics Corporation

H. C. Cano, Jr., D. J. Boldjak, M. A. Goode, R. M. Bond, C. G. Anderson, R. C. Robertson

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

Prepared:

ROBERT B. PARKER
Project Engineer

APPROVED:

JOHN A. BEYERLE, Lt Colonel, USAF
Acting Chief, Command & Control Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COAR) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-3, Vol II (of three)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY System/Subsystem Specification		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 6 Jan 81 - 30 Sep 82 6. PERFORMING ORG. REPORT NUMBER DMA-2-014
7. AUTHOR(s) H.C. Conn, Jr. R.M. Bond D.J. Rodjak C.G. Anderson M.A. Goode R.C. Robertson		8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0039
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Dynamics/DSD/Central Center North Grant Lane Ft Worth TX 76108		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63701B 32050326
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE January 1983 13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Roger Panara (COEE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Engineering
programming environment
software tools

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Vol I (of three) describes the development of the design and supporting documentation for an incremental and evolving integrated modern engineering software production environment for the Defense Mapping Agency.

Vol II is the System/Subsystem Specification.

Vol III is the Functional Description.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SYSTEM/SUBSYSTEM SPECIFICATION
TABLE OF CONTENTS

Section	Title	Page
1.	GENERAL	4
1.1	Purpose of the System/Subsystem Specification	4
1.2	Project References	5
1.3	Terms and Abbreviations	7
2.	SUMMARY OF REQUIREMENTS	8
2.1	System/Subsystem Description	8
2.2	System/Subsystem Functions	11
2.2.1	Accuracy and Validity	12
2.2.2	Timing	12
2.3	Flexibility	12
3.	ENVIRONMENT	14
3.1	Equipment Environment	14
3.1.1	VAX Environment	14
3.2	Support Software Environment	20
3.2.1	VAX Software Tools	27
3.2.1.1	DMATRAN(IFTRAN)/FORTRAN 77/COBOL 74	27
3.2.1.2	USE.IT	28
3.2.1.3	SDDL	30
3.2.1.4	IS/1	33
3.2.1.4.1	INed	39
3.2.1.4.2	INword	39
3.2.1.4.3	SCCS	43
3.2.1.5	FAVS/RXVP80	44
3.2.1.6	CAVS	46
3.2.1.7	VUE	48
3.2.2	General Support	49
3.2.2.1	HYPERGRAPHICS	49
3.3	Interfaces	52
3.4	Security and Privacy	52
3.5	Controls	52
4.	DESIGN DETAILS	54
5.	DISTRIBUTION AND ADDRESSEES	55

SYSTEM/SUBSYSTEM SPECIFICATION
FIGURES

Number	Title	Page
2.1	Near-Term System Configuration for DMA Modern Programming Environment	9
3.1	VAX-11/780 System Configuration	14
3.2	Technical Specifications for VAX-11/780 Processor	15
3.3	MPE Project Management Overview	21
3.4	MPE Usage Scenarios Overview	23
3.5	MPE Usage Scenarios	24
3.6	A Life Cycle Model Scenario Employing USE.IT	29
3.7	Hierarchical Data Development and Communication	30
3.8	SDDL Software Design Process	32
3.9	IS/1 Commands and Programs	36
3.10	INword Features	41
3.11	INword Utilities	42
3.12	Steps in Validating a Program with FAVS or RXVP80	44
3.13	Software Analysis and Testing Augmented by FAVS or RXVP80	45
3.14	CAVS Use in Developing Systems	47
3.15	HYPERGRAPHICS Command List	50

SECTION 1. GENERAL

1.1 Purpose of the System/Subsystem Specification. This report is the System/Subsystem Specification, contract data requirements list (CDRL) item A007, produced as part of the Interactive Computer Program Development System Study for the Defense Mapping Agency (DMA).

The purpose of this document is to provide a technical description of the components of the recommended near-term DMA modern programming environment (MPE). (See Final Report for a complete explanation of the evolution of the near-term MPE recommendation.) This document includes descriptions of the DMA MPE tool bearing host (TBH) and software life cycle tool support environment for the near-term MPE configuration. The tools described are the ones which best satisfied the requirements and constraints of the DMA environment at the time this document was produced. The implementation of this system should have the recommended tools or equivalent tools which satisfy the criteria and constraints of the DMA MPE. Only the FORTRAN and COBOL languages are addressed in this document.

1.2 Project References.

a. FEDSIM (Federal Computer Performance Evaluation and Simulation Center) Installation Review - DMAHTC, November 1980

b. FEDSIM Installation Review - DMAAC, August 1980

c. FEDSIM Optimization and Error Rate Studies, February 1981

d. Statement of Operation Need and System Operational Concept, CDRL A002 for contract no. F30602-81-C-0039 - Interactive Computer Program Development System Study, February 1982

e. Tool Evaluation Plan, CDRL A003 for contract no. F30602-81-C-0039 - Interactive Computer Program Development System Study, September 1981

f. Tool Survey, CDRL A004 for contract no. F30602-81-C-0039, Interactive Computer Program Development System Study, February, 1982

g. Alternative Analysis, CDRL A005 for contract no. F30602-81-C-0039, Interactive Computer Program Development System Study, March, 1982

h. Functional Description, CDRL A006 for contract no. F30602-81-C-0039, Interactive Computer Program Development System Study, March, 1982

i. RADC-TR-78-268 Volume II (of three) FAVS Fortran Automated Verification System Users Manual

j. CR-2-970 CAVS COBOL Automated Verification System Functional Descripion, November, 1980

k. A Microcomputer Based Classroom Lecture System By Thomas C. Irby and Darrell Ward, North Texas State University, Dept. of Computer Science

l. Interactive Systems Corporation:

   1) IS/1 Workbench Programmers Manual for VAX/VMS

   2) IS/1 Workbench Users Guide for VAX/VMS

   3) IS/1 Text Processing Manual

4) Interactive Systems Corporation Software Product Descriptions

m. Digital Equipment Corporation (DEC):

1) VAX Architecture

2) VAX Hardware Handbook

3) VAX Software Handbook

n. Higher Order Software (HOS), Cambridge, MA. Product Descriptions:

1) USE.IT

o. Science Applications, Inc (S.A.I.):

1) SDDL

p. General Research Corporation, Santa Barbara, CA. Users Manuals.

1) IFTRAN

2) RXVP80

1.3 Terms and Abbreviations.

ANSI	AMERICAN NATIONAL STANDARDS INSTITUTE
APL	A PROGRAMMING LANGUAGE
ASCII	AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE
CDRL	CONTRACT DATA REQUIREMENTS LIST
CPU	CENTRAL PROCESSING UNIT
CRT	CATHODE RAY TUBE
DCL	DIGITAL COMMAND LANGUAGE
DDP	DISTRIBUTED DATA PROCESSING
DEC	DIGITAL EQUIPMENT CORPORATION
DMA	DEFENSE MAPPING AGENCY
DMAAC	DEFENSE MAPPING AGENCY AEROSPACE CENTER
DMAHQ	DEFENSE MAPPING AGENCY HEADQUARTERS
DMAHTC	DEFENSE MAPPING AGENCY HYDROGRAPHIC/ TOPOGRAPHIC CENTER
DNA	DIGITAL NETWORK ARCHITECTURES
FAVS	FORTRAN AUTOMATED VERIFICATION SYSTEM
FCN	FUNCTION
FEDSIM	FEDERAL COMPUTER PERFORMANCE AND EVALUATION AND SIMULATION CENTER
FTN77	FORTRAN 77
GD/DSD	GENERAL DYNAMICS/DATA SYSTEMS DIVISION
HOL	HIGH ORDER LANGUAGE
HOS	HIGHER ORDER SOFTWARE
IS/1	INTERACTIVE SYSTEMS/ONE
LAN	LOCAL AREA NETWORK
MB	MEGABYTE
MGT	MANAGEMENT
MPE	MODERN PROGRAMMING ENVIRONMENT
PDAY	PARTIAL DAY
PWB	PROGRAMMER'S WORK BENCH
RADC	ROME AIR DEVELOPMENT CENTER
RAT	RESOURCE ALLOCATION TOOL
RJE	REMOTE JOB ENTRY
ROM	READ ONLY MEMORY
SCCS	SOURCE CODE CONTROL SYSTEM
SDD	SOFTWARE DESIGN AND DOCUMENTATION
SDDL	SOFTWARE DESIGN AND DOCUMENTATION LANGUAGE
TBH	TOOL BEARING HOST
UNIX	TRADEMARK OF BELL LABORATORIES (OPERATING SYSTEM)
VMS	VIRTUAL MEMORY SYSTEM
WDCS	WRITABLE DIAGNOSTIC CONTROL STORE

SECTION 2. SUMMARY OF REQUIREMENTS

2.1 System/Subsystem Description. The near-term system design was developed, through the process described in Section 15 of the Final Report , to meet the immediate needs of DMA. The near-term MPE is based upon a VAX used as a front-end software development environment to a production UNIVAC mainframe as described in Figure 2.1. This configuration provides a software development capability with minimum schedule and technical risk at low cost. As defined, the system has a high probability for improving productivity. The MPE supports all life cycle development phases and the maintenance and project management functions. In this document 'maintenance functions' is defined as post production software development activity requiring work in one or more phases of the life cycle: requirements, design, programming, and testing. These would include activities such as the correction of software errors discovered in production programs and modifications or upgrades to programs already on production status.

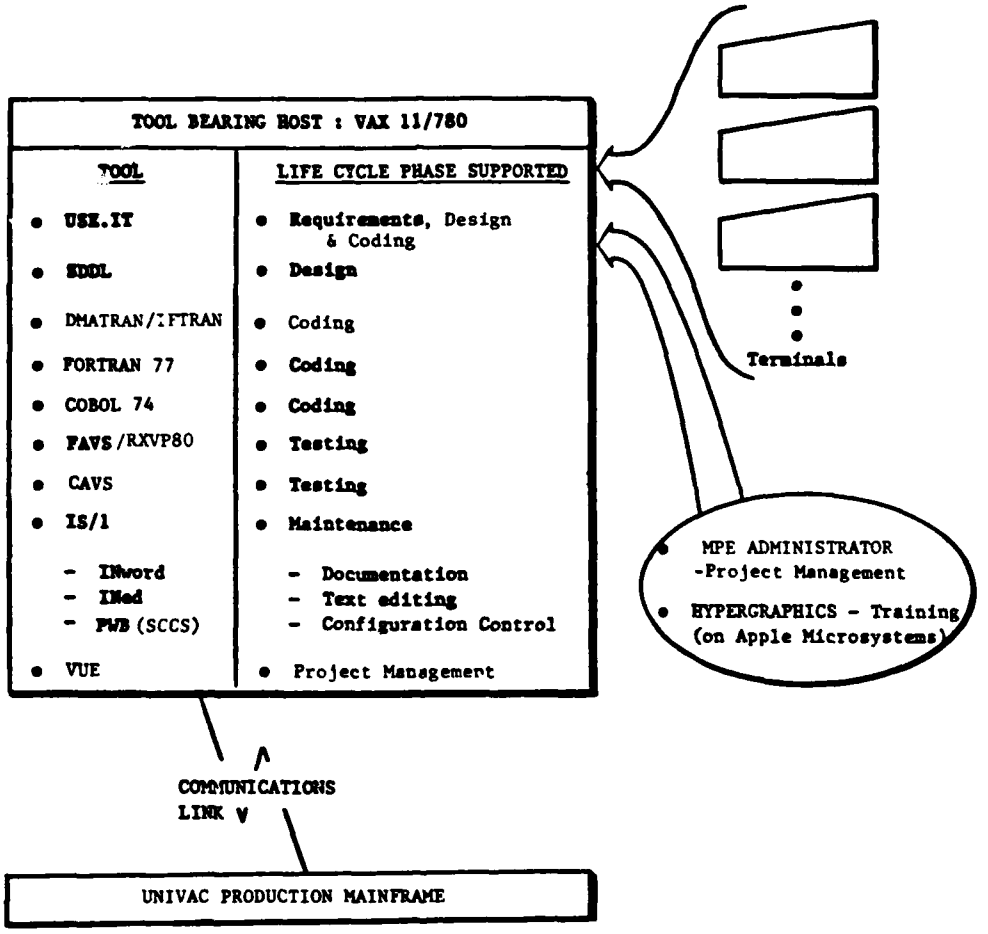


Figure 2.1 Near-Term System Configuration for DMA Modern Programming Environment

A VAX-11/780 will be utilized to host a FORTRAN and COBOL software development system to support the requirements, design, programming, testing and maintenance phases of the software development life cycle as well as project management. An overview of the specific support provided by this proposed system is given in the following paragraphs.

The AXES portion of the USE.IT tool will functionally support the specification and analysis of the requirements of programs. Certain categories of programs will also be developed within the USE.IT life cycle and supporting library. The design of the remaining categories of programs will require the generation of a software design document through the use of the Software Design and Documentation Language (SDDL). Definitions of the 5 categories of software development and their associated MPE usage scenarios are found in Section 3.2 of this report.

All coding will be accomplished in DMATRAN(or its commercial version IFTRAN), ANSI X3.9-1978 FORTRAN (77) or ANSI X3.23-1974 COBOL (74). FORTRAN 77 and COBOL 74 will serve as a common interface between the VAX software development machine and the production UNIVAC mainframe to which source code of completed programs will be sent for final compilation and production status. Testing and optimization of programs will be performed on the VAX using the FORTRAN Automated Verification system (FAVS), or its commercial version RXVP80, for FORTRAN 77 programs and the COBOL Automated Verification System (CAVS) for COBOL 74 programs. Additionally on the VAX will reside a configuration of the Interactive Systems/One (IS/1) system supporting documentation, text editing, and the configuration control function of the maintenance phase.

Additional non-hosted support will be required. The MPE administrator and toolsmith functions will support the project management function as well as system management; and HYPERGRAPHICS, a tool for building presentations, lectures or interactive lessons will be utilized for training purposes. The selection of HYPERGRAPHICS is based on its simplicity in use as well as cost and availability. The generation of visual material to support a training document is easily performed and maintained.

The VAX and UNIVAC computers will need to be connected through a communications link described in Sections 2.2 and 3.3. To support users in a timely manner and to provide adequate access, multiple VAX computers will be required. A recommendation of three identically configured systems at each center is explained in Section 16.2 of the Final Report. Additionally, the VAX's within each center will be connected

through the DECnet Local Area Network (LAN) thus providing intra-system communications and back-up capabilities for the development systems.

2.2 System/Subsystem Functions. The near-term MPE functions as a tool to provide life cycle support to the software development process. In the following paragraphs the individual functions of each major element will be described as well as the function of the aggregate.

A VAX-11/780 computer will be utilized to support requirements specifications, design, coding (data entry and analysis), documentation, testing, configuration control, and project management. The computer will support multiple terminals over a range of geographic locations, depending upon the communication and protocol utilized.

The specification and documentation of the requirements of a computer program will be partially automated through the use of USE.IT. This tool will allow the interactive development of a requirements specification document using a defined methodology, and analysis of the specification for data flow and control sequences. When the program specified can be categorized to fit within certain constraints, other facilities of the USE.IT tool can be used which will directly generate a high order language (HOL) program to accomplish the specified task.

The design of remaining programs will be accomplished utilizing SDDL. Whether the task is accomplished by an individual or a team, the tool will provide precise, accurate and orderly transitions between requirements, design and coding activities as well as intra-design activities. The tool provides, through a prescribed methodology, the capability to describe the design in simple, understandable constructs; allow for checking of the design constructs; and translate the design into a readable design document.

Data entry will be performed interactively when generating new code or documentation. This activity will be supported by the state-of-the-art word processing and text editing capabilities of the IS/1 Programmer's Work Bench (PWB). When new code is being generated, a compiler must be resident for syntactical and semantic analysis prior to the test and integration phases This will require FORTRAN 77 and COBOL 74 compilers to be resident on all VAX's.

FAVS(or RXVP80) and CAVS will provide static and dynamic analysis of the specified HOL source code including usage, path flow and coverage statistics. Additional capabilities

to enhance documentation, such as the output of cross-reference tables and summary data or pretty-printing the source input, will also be included. Note that RXVP80 is the commercial version of FAVS. Though they possess the same basic capabilities as outlined above, there are differences.

Configuration control will be supported through the use of SCCS. The configuration management of HOL code and support documentation, including on-line requirements and design information and test data, will be provided.

Project management will be supported through the use of VUE. This tool performs resource allocation and analysis, time and cost analysis, and report processing.

Due to the complexity of the proposed near-term environment, the evolutionary process required to achieve the far-term environment, and a need for a focal point for identification/resolution of problems, support tools must be provided outside the development environment.

The MPE administrator and toolsmith functions will be staffed positions primarily serving as the focal point for management to observe the system activities and as a source of information for MPE training activities. Tasks will include performing error rate studies, helping programmers with MPE usage questions/problems and the identification of needs not satisfied within the user/management communities.

HYPERGRAPHICS, an off-line training tool, will be used to support training functions. This will provide low cost training to personnel outside the production environment.

Communication links will be established between the VAX 11/780's and mainframe computer. The VAX's will also be linked through a DECnet LAN.

2.2.1 Accuracy and Validity. This section does not apply to this specification.

2.2.2 Timing. This section does not apply to this specification.

2.3 Flexibility. This section does not apply to this specification.

SECTION 3. ENVIRONMENT

3.1 Equipment Environment. This section provides a description of the new equipment required for the operation of the near-term MPE.

3.1.1 VAX Environment. The VAX equipment environment will consist of three VAX-11/780's, as specified in the Final Report and described in Figure 3.2. Each system will have interactive terminals (INtext II's and VT100's, with Retrographics) located within 1000 feet of the host machine and multiport memory consisting of 2 megabytes of memory. Additional recommendations are eleven removable disk packs and 50 tapes per VAX. The system configuration (SV-AXDBC-CA with additions) is presented in Figure 3.1 and will require 70 square feet of space for system cabinets and the console terminal.

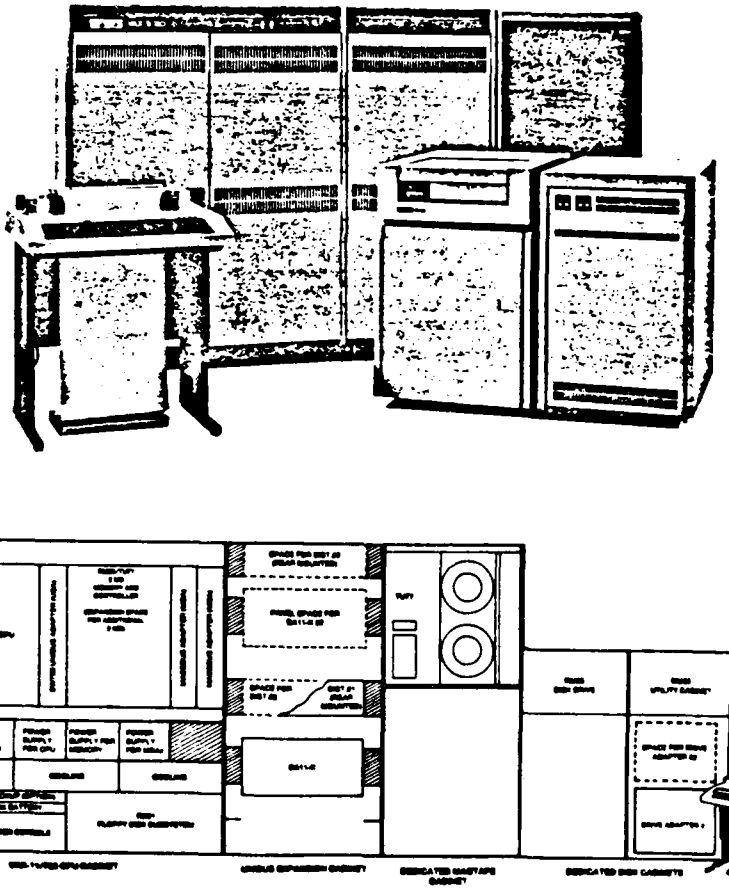


Figure 3.1 VAX-11/780 System Configuration

Central Processing Unit	
Processor Type	Microprogrammed, 99-bit control store word Microcontrol store instruction time 200 nanoseconds Control store size - 5K words (99-bit words), 4K words ROM, and 1K words WDCS
Internal data path	32 bits
CPU Cache Memory	
Size	8K bytes, bipolar with parity
Effective main memory cycle time	1800 nanoseconds/64 bits
Typical hit ratio	95%
Typical cache cycle time	290 nanoseconds
CPU Address Translation Buffer	
Size	128 address translations
Typical hit ratio	97%
CPU Clocks	
Real-time clock	Crystal controlled, .01% accuracy 1-microsecond resolution
Time-of-year clock	Includes recharging battery backup for over 100 hours
VAX Instruction Set	16 32-bit registers 248 basic operations 32 priority interrupt levels
Multiple data types	Integer, floating point, packed decimal, character string, variable bit fields, and numeric strings PDP-11 compatibility mode instructions
Addressing modes	9

Figure 3.2 Technical Specifications for VAX-11/780 Processor (page 1 of 5)

Other features	Power fail/automatic restart, Single serial line ASCII console interface, 8-line communications multiplixers DZ-11A, DZ-11B, DZ-11E; RX01 floppy disk drive with LSI-11 microcomputer, Writable Diagnostic Control Store (WDCS), Serial line unit for remote diagnostics, Virtual console commands from LA120 terminal
Main Memory	
Physical address lines	1 billion bytes (30 bits)
Physical expansion	8 megabytes in 256K-byte increments
Parity	8-bit error correcting code (ECC) per 64-bit quadword
Technology	16K-bit dynamic RAMs (200 nanosecond access time)
Cycle times	800 nanoseconds per 64-bit read (1300 nanoseconds with single-bit errors) 1400 nanoseconds per 64-bit write
Power failure protection	Optional battery backup
I/O UNIBUS Adapter (1 standard)	
Maximum UNIBUS I/O rate	1.5 Mb/sec through buffered data paths
Buffered data paths	15 total, 8-byte buffer in each
Maximum number of bus loads	18 without a repeater
Interrupts	Directly vectored via UNIBUS adapter

Figure 3.2 Technical Specifications for VAX-11/780 Processor (page 2 of 5)

Memory Battery Backup	
Minimum backup time	10 minutes
Electrical Power Requirements	
AC line voltage	120/208V
Frequency tolerance	59-61 Hz
Phases	3 phase phase A: 11.2A max. continuous phase B: 9.9 A max. continuous phase C: 13.1 A max. continuous neutral: 14.4 A max. continuous
AC cable length	3 m (9.84 ft.) from back of cabinet
Maximum ac power consumption	6225 watts
Typical Power Requirements and Thermal Dissipation	
Fully configured multiport memory	1800 watts 6140 BTU/hr. 1550 kcal/hr.
Environment	
Operating:	
Temperature	15° to 32°C (59° to 90°F)
Relative humidity	20% to 80%
Nonoperating:	
Temperature	-40° to 66°C (-40° to 151°F)
Relative humidity	0 to 95%

Floating Point Accelerator
Enhances performance of all floating point instructions (single and double precision) including polynomial evaluation, integer/floating conversions, 8-, 16-, and 32-bit integer multiply.

Figure 3.2 Technical Specifications for VAX-11/780 Processor (page 3 of 5)

Peripherals

TEU77 tape transport device:

Program selectable, 800/1600 bpi, 9-track, 125 in/sec automatic loading magnetic tape transport.

INtext II terminal:

INtext II is the INTERACTIVE, Inc. text editing terminal based on the Perkin-Elmer 1251 terminal. The terminal features a video display of 24 lines with 80 characters each. ASCII text and graphic symbols may be displayed, and a line-drawing option is provided. The screen is 12 inches along the diagonal and is hooded to improve readability. INtext II has a tiltable screen and a detachable keyboard. The terminal can communicate with a computer at speeds ranging from 75 to 9600 baud. INtext II is configured with special microcode to optimize operation with the text editor INed. Many of the functions normally performed by the host CPU are done locally in the INtext II microcode; allowing the host CPU to support more editing stations. INtext II may also be used as a standard ASCII terminal. Terminal parameters can be set for ANSI, VT100, VT52, or dumb modes. INtext II has labeled function keys corresponding to specific functions found in INTERACTIVE's screen editor, INed.

VT100 terminal (with Retrographics VT640):

The VT100 is a high performance video display terminal which provides maximum flexibility. SET-UP features including scroll mode, auto repeat, background, cursor style, margin bell, key click, and 80 or 132 columns allow the terminal to be configured to operator preference and provide compatibility with the host computer.

The Retrographics VT640 enhancement, a product of Digital Engineering, requires no software modifications and delivers full graphics capabilities without diminishing the features of the DEC VT100.

REM05 disk drive:

The REM05 is a single access 256MB removable disk pack drive with one disk pack supplied. The transfer rate is 1.2MB/sec peak with an average access time of 38.3 msec.

Figure 3.2 Technical Specifications for VAX-11/780 Processor (page 4 of 5)

DUP11 communications device:

The DUP11 is a single-line, program controlled, double-buffered communications device. The self-contained unit is capable of handling a wide variety of protocols, byte and bit oriented. The DUP11 can also be used in conjunction with customized code for unique applications.

MUX 200/VAX multiterminal emulator:

MUX200/VAX is a software package that allows data files to be transferred by emulating CDC's 200UT Mode 4A communications protocol. User's may choose ASCII or BCD character codes for transmission. Using interactive terminals, users can communicate with the host system at command level or can send or receive batch processing jobs. The VAX-11 terminals appear as 200UT terminals to the host computer. Users communicate with the host computer over a single synchronous communications line capable of operating at speeds up to 9600 bits per second. The host computer views the MUX200/VAX line as 16 multidropped 200UT terminals. Features include detection of user-defined strings that permit the VAX-11 to spool output data received from the host system to a line printer; transmission of up to 8 data files to the host with a single command; and the capability to support local data processing on the VAX while operating independently of the comunications link.

DECnet-VAX

DECnet-VAX allows a suitably configured VAX/VMS system to participate as a Phase III DECnet node in computer networks. DECnet-VAX offers task-to-task communications, network file transfer, adaptive path routing, network management, and network resource access capabilities (including Network Command Terminals) using the DIGITAL Network Architecture (DNA) protocols. DECnet-VAX communicates with adjacent and non-adjacent Phase III nodes over synchronous communication lines.

Figure 3.2 Technical Specifications for VAX-11/780 Processor (page 5 of 5)

3.2 Support Software Environment. This section provides a description of the support software with which the user of the near-term MPE must interact. This includes descriptions of the various categories of software development existing at DMA and how each would interact with the recommended MPE.

All software development is monitored through the use of the project management tool, VUE. Examples of the inputs and outputs of VUE are illustrated in Figure 3.3. Upon receiving a job request, VUE is initiated for the job, and at various points in the scenarios, VUE is updated to reflect pertinent decisions and actions.

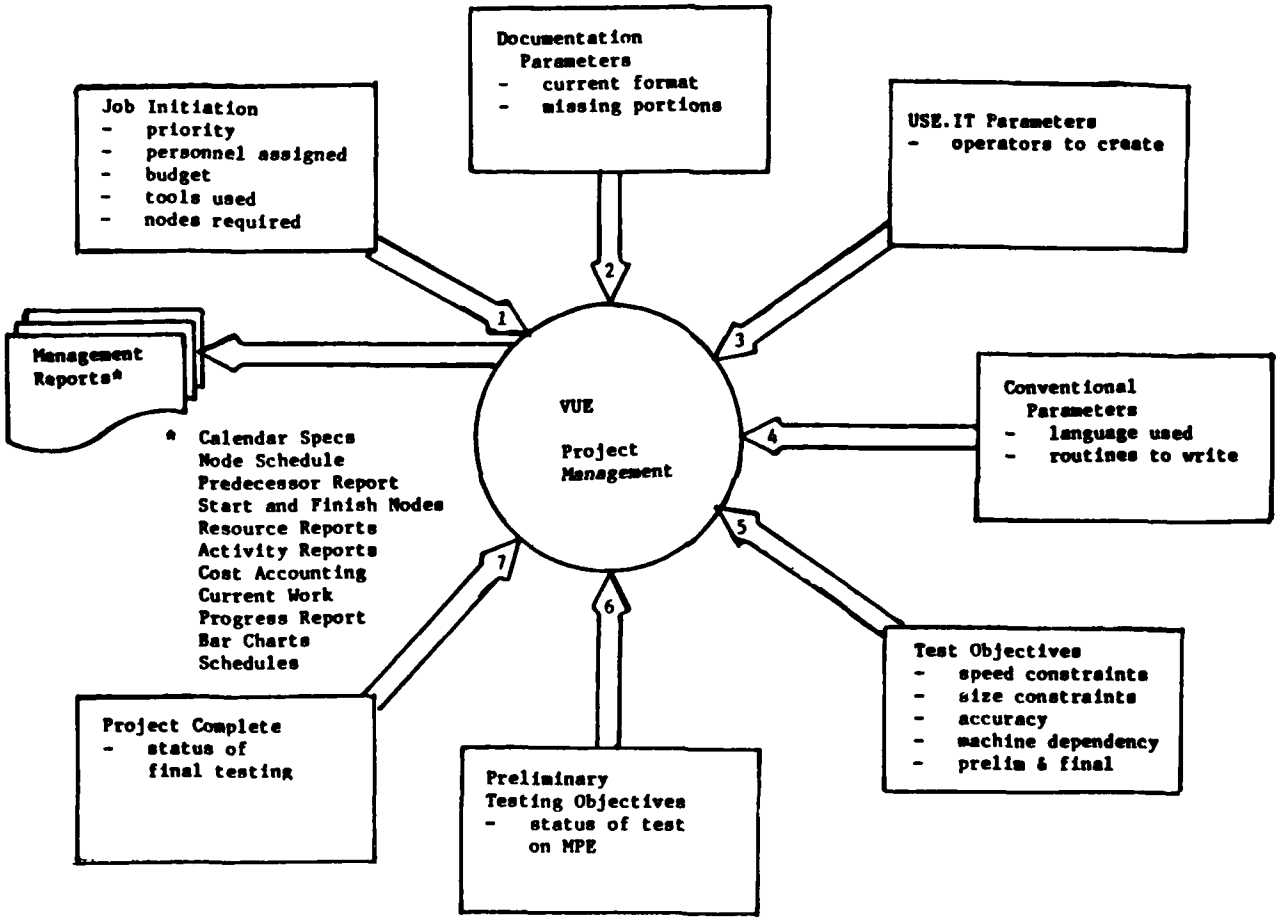


Figure 3.3 MPE Project Management Overview

The usage of tools in the MPE is best discussed in terms of scenarios. For purposes of discussion, scenarios will be considered for the following categories of software development:

1. maintenance of existing software which has not been upgraded through the Software Improvement Program (SIP), (Part of this program consists of an effort to improve existing UNIVAC software.)

2. maintenance of existing software which has been SIP upgraded,

3. software under development, for which standards were not specified

4. new software to be developed by DMA, for which standards are to be specified, and

5. new software to be developed by contractor, for which standards are to be specified.

The techniques discussed are intended to demonstrate the applicability of the recommended tools to the various scenarios. Specific usage methodologies will be developed during the MPE system implementation as outlined in Section 19.1 of the Final Report.

Application of the MPE tools to the DMA software scenarios is illustrated in Figures 3.4 and 3.5.

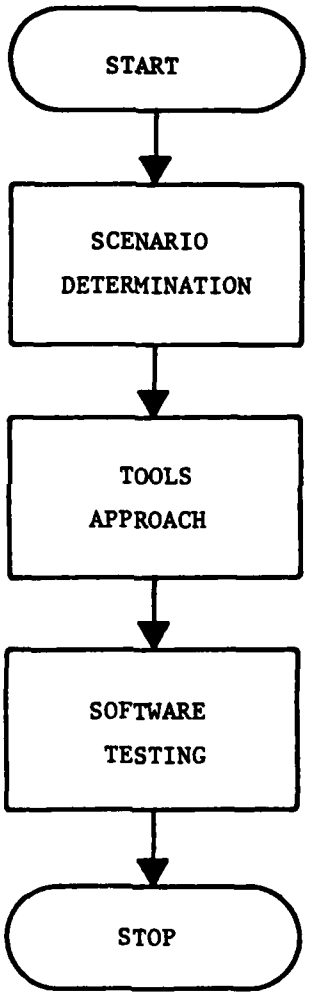


Figure 3.4 MPE Usage Scenarios Overview

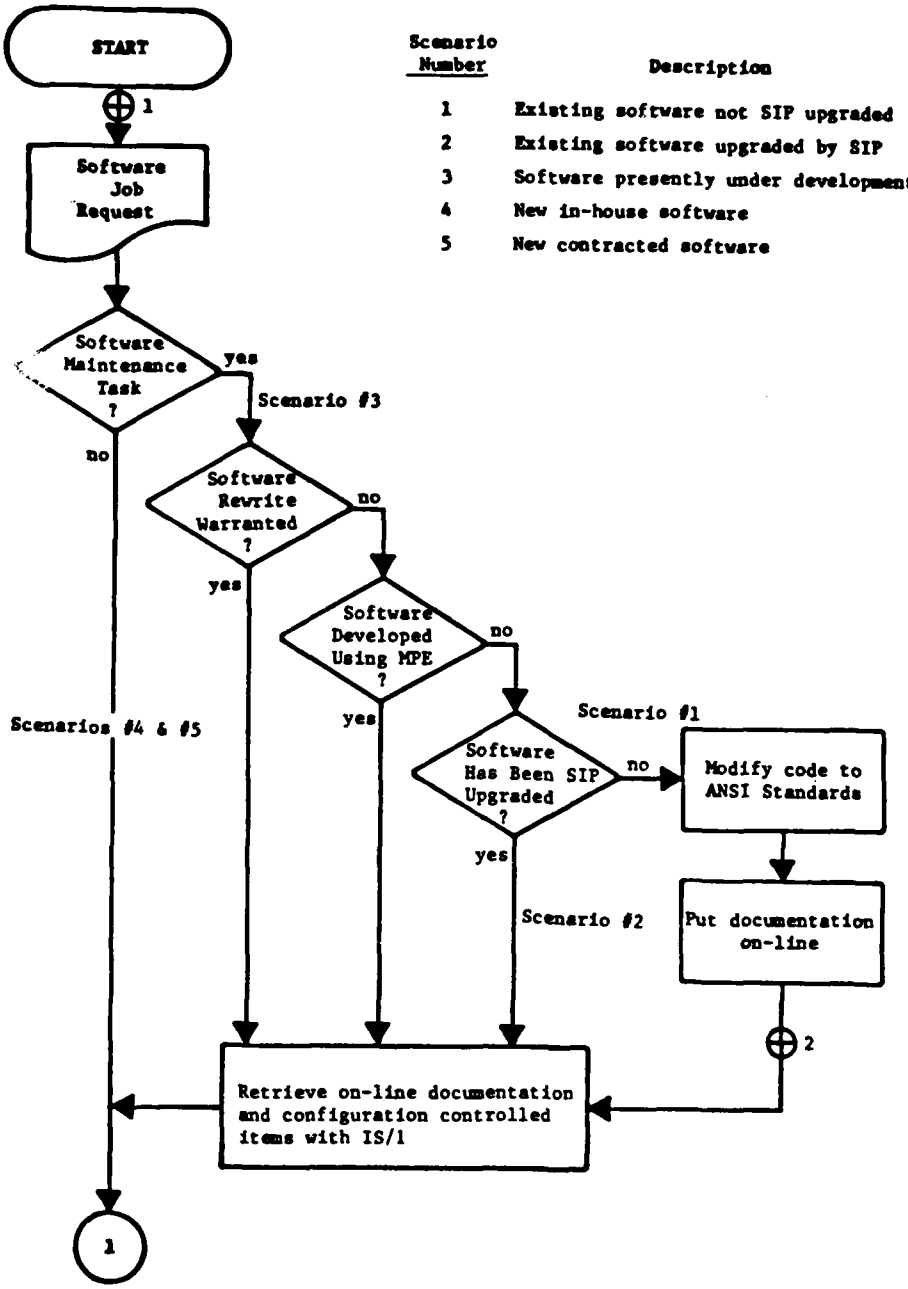


Figure 3.5 MPE Usage Scenarios
(page 1 of 2)

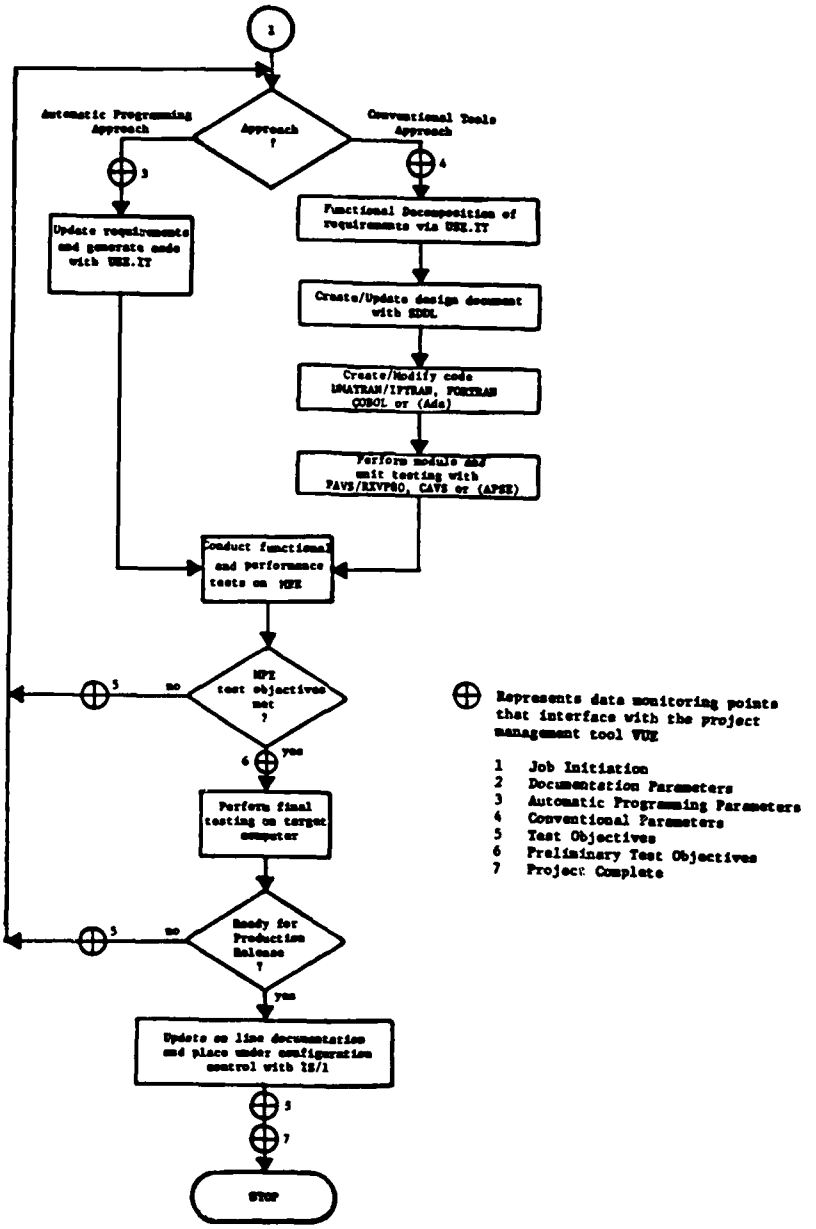


Figure 3.5 MPE Usage Scenarios
(page 2 of 2)

Upon receiving a job request, the project management tool, VUE, is initiated for the job and at various points in the scenarios, the project management system is updated to reflect pertinent decisions and actions.

Within the defined scenarios, one of two basic tool approaches will be followed.

The first, referred to as the "automatic programming approach", will make repeated use of the subsets of the tool USE.IT until performance criteria are achieved. The usage of the various subsets is as follows:

- the USE.IT graphics editor is used to enter program structures, called control maps, to functionally decompose requirements and design specifications as well as changes, if any, which are required as a result of performance testing,

- the Analyzer verfies internal consistency and interfaces,

- the Resource Allocation Tool (RAT) automatically produces programs from Analyzer output,

- source produced by the RAT is compiled and linked, and

- the system is performance tested to determine acceptability.

Failure to pass performance testing results in repetition of these steps until criteria are satisfied.

There appears to be no restriction on the size of system which may be developed with USE.IT. As systems are developed via USE.IT, generic operations are developed and placed in a library for use as building blocks on subsequent systems. For this reason, detailed documentation within AXES statements is considered mandatory.

The second, referred to as the "conventional tools approach", will make use of the USE.IT, SDDL, DMATRAN/IFTRAN or FORTRAN or COBOL, and FAVS/RXVP80 or CAVS tools through the life cycle. Utilization of tools in the "conventional tools approach" consists of repeated application of the following procedures until performance criteria are achieved.

- The USE.IT graphics editor is used to functionally decompose requirements specifications.

- SDDL is used to originate the design or make design changes, if any, which were mandated as a result of performance testing.

- Source code (DMATRAN/IFTRAN, FORTRAN or COBOL) is modified to reflect changes brought about by design changes, performance changes, or FAVS/RXVP80 or CAVS evaluation.

- FAVS/RXVP80 or CAVS are envoked to detect syntax errors, perform static analysis, and perform execution analysis.

- Performance testing is evaluated to establish the acceptability of the system. Failure to pass performance testing results in repeating the process.

One of these tool application approaches is followed until the preliminary test objectives are met. At this time, the source is transmitted via data link to the target host for final testing.

While testing on the target host, the project management system is apprised of the test status. Upon successful completion of final test objectives, job completion data is processed by the project management system. This action prevents the system status from being obscured from control and insures a match between production software and the associated documentation. Target host test objectives will verify proper usage of machine dependent devices, software and techniques. Once final testing is completed and the system is ready for production status, on-line documentation such as requirements and design documents, source code and test data should be updated and placed under configuration control using SCCS of the IS/1 PWB.

3.2.1 VAX Software Tools. This section provides descriptions of the VAX hosted software tools in the Near-Term MPE.

3.2.1.1 DMATRAN(IFTRAN)/FORTRAN 77/COBOL 74. Coding will be accomplished in DMATRAN(IFTRAN), FORTRAN 77 or COBOL 74. DMATRAN(or its commercial version IFTRAN) is an extension to FORTRAN that allows the use of the SEQUENCE, DOWHILE, DOUNTIL, IFTHENELSE, and CASE control constructs. The DMATRAN(IFTRAN) precompiler translates the DMATRAN(IFTRAN) statements into standard FORTRAN while passing all other statements unchanged to a file which can then be compiled by a FORTRAN compiler. In addition to the translation, the precompiler checks the control structure for proper use of

DMATRAN(IFTRAN) control structures and issues error messages if violations occur.

The precompiler provides the following additional features to improve code production:

1. Indented listing of the DMATRAN(IFTRAN) source code.

2. Editing functions which include in-line comments, double-spacing around comments, indentation control, selective page ejection, and selective suppression of the source listings.

FORTRAN 77 refers to the American National Standard Programming Language FORTRAN, ANSI X3.9-1978 approved by the American National Standards Insititute (ANSI) on April 3, 1978. This is the FORTRAN version currently in use (with extensions) at both DMAAC and DMAHTC on their UNIVAC equipment and supported on VAX-11/780 equipment. COBOL 74 refers to the American National Standard Programming Language COBOL, ANSI X3.23-1974 approved by ANSI.

3.2.1.2 USE.IT. Problem definition always occurs prior to requirement specification, however, analysis of solvability rarely occurs. One reason behind this fact is the labor intensive characteristic of feasibility studies. A software tool, USE.IT, automates this process. The USE.IT software system is an integrated set of tools for automating the software lifecycle development process. Implementation of a system is performed through the use of a user-selected syntax language modeled with the Higher Order Software AXES language interface. The user specifies the system components and dataflow in a selected syntax; the AXES subsystem performs an analysis of the specification; and the Resource Allocation Tool (RAT) generates a FORTRAN 66 program representing the system. Future expansion of the USE.IT system is planned to include the capability to automatically generate code for FORTRAN 77, COBOL, and Ada, from a single specification. Figure 3.6 illustrates the typical software lifecycle and the software lifecycle as it would be if USE.IT were employed.

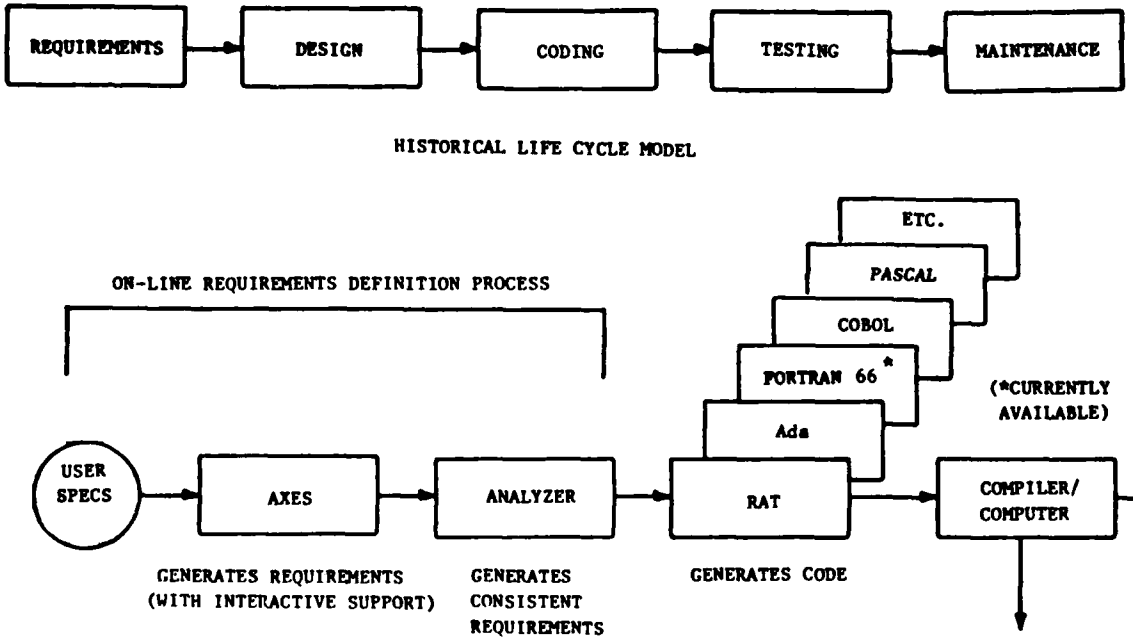


Figure 3.6 A Life Cycle Model Scenario Employing USE.IT

3.2.1.3 SDDL. The Software Design and Documentation Language(SDDL) is a software support tool used to partially automate the generation and checking of a software design document, (SDD). An SDD is needed because a computer programming language is a satisfactory communications medium for only a few links in the software development process; programmer-to-programmer and programmer-to-machine. Other links in the process include designer-to-programmer, designer-to-designer, designer-to-manager, manager-to-customer and customer-to-designer. This is a complex network of information flow which usually requires multiple mediums of communication. SDDL was developed by the Jet Propulsion Laboratory to provide a single communication medium that would be usable over all links except programmer-to-machine (see Figure 3.7).

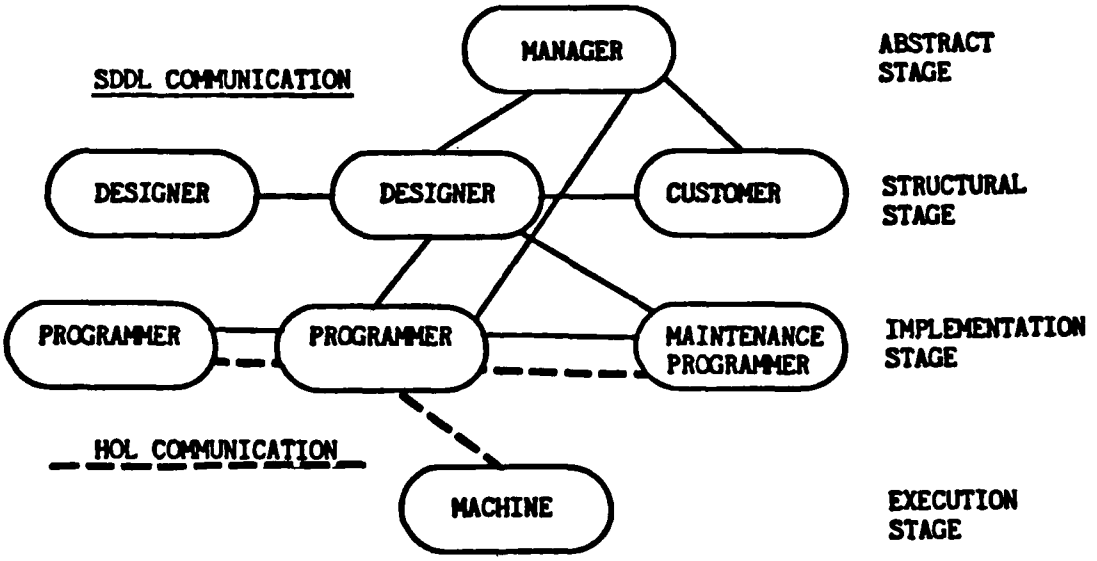


Figure 3.7 Hierarchical Data Development and Communication

The objective of SDDL is to provide an effective communications medium to support the design and documentation of complex software applications. This objective is met by providing (1) a processor which can convert design specifications into an intelligible, informative machine-reproducible document, (2) a design and documentation language with forms and syntax that are simple, unrestrictive, and communicative, and (3) a methodology for effective use of the language and processor. The processor has the capability to format documents, summarize design information in the form of reports and handle various user-controlled directives.

SDDL is accessed interactively, but is a batch oriented process. It is used to periodically generate a SDD which will reflect the current status of the design process. Information may be generated on variable usage, program modules, changes to programs, management information and many other states of the design as desired by the project team members. A programming language's structure and keywords may even be modeled to an extent that very little effort would be required to change the final SDD into source code.

The source language for SDDL was originally Simscript II.5, which was only available on large computers. Release version four has been rehosted to a Harris computer using PASCAL as the source and is rehostable to any computer system supporting Jensen and Wirth standard PASCAL which is supported by the VAX.

Utilization of SDDL is accomplished through a language syntax that is simple and flexible. The SDDL processor reads a text file phrased in the language, then reformats the file by providing indentation, control flow lines, and user specified cross reference tables. The printed SDD contains the reformatted input, a table of contents, and module hierarchy reports. The indentation can be modeled by using the structural keywords of a specific language. The exploitation of these and other features of SDDL provides a vehicle for establishing standards and conventions in the design process. Figure 3.8 provides a graphic illustration of the SDDL software design process.

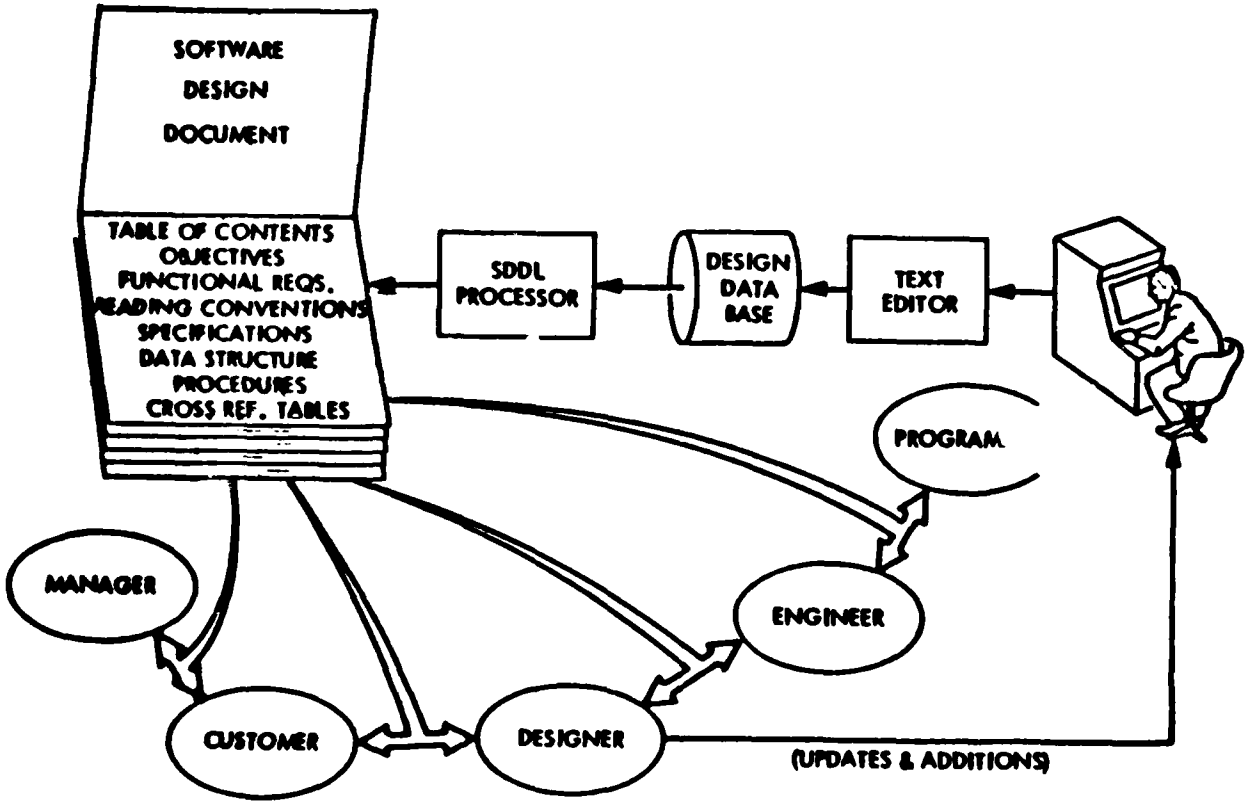


Figure 3.8 SDDL Software Design Process

3.2.1.4 IS/1. The IS/1 Workbench for the VAX is a facility that provides a convenient working environment and a uniform set of tools for computer program development, document preparation and text processing. It is a general-purpose, multi-user, interactive system based on Bell Laboratories' PWB/UNIX system specifically engineered to make the designer's, programmer's and documenter's environment simple, efficient, flexible and productive. The system runs on Digital Equipment Corporation's VAX-11/780 computer system as a subsystem of the VAX/VMS operating system. It contains features such as:

- o A hierarchical file system
- o A flexible, easy-to-use command language that can be tailored to meet specific user needs
- o The ability to execute sequential, asynchronous and background processes
- o A line-oriented context editor
- o A flexible document preparation and text processing system
- o A high-level programming language conducive to structured programming (C).
- o A variety of system programming tools
- o The integration of the VAX/VMS and UNIX environments, such that the look and feel of UNIX is preserved while allowing access to VMS when desired.

Because the IS/1 Workbench for the VAX runs as a subsystem of the VAX/VMS operating system provided by DEC, full access to DEC software is available at all times. In particular, the Workbench tools may be used to develop native mode VAX/VMS programs. Compiled programs adhere to VAX object linkage and calling sequence conventions. The C compiler, though not recommended in the near-term environment, is provided with the Workbench system, generates VAX native-mode code and obeys VAX stack conventions. Moreover, the Workbench Command Interpreter is capable of invoking native DEC utilities such as the linker and FORTRAN, as well as Workbench utilities with the same command language. Most Workbench programs can be initiated directly by using the VAX/VMS DCL command interpreter as well as by using the Workbench command interpreter. The file system provided by VAX/VMS is fully available to the Workbench subsystem. This file system consists of directories and files arranged in a hierarchical structure. Some of its features include:

- o Simple and consistent naming conventions. Names can be absolute or relative to any directory in the file system hierarchy.

- Automatic file space allocation and deallocation that is transparent to users.

- A flexible set of directory and file protection modes. All combinations of read, write and execute access are allowed independently for the owner of each file or directory, for a group of users (e.g., all members of a project) and for all other users. Protection modes can be set dynamically.

- Facilities for creating, accessing, moving and processing files, directories or sets of these in a simple, uniform and natural way.

- Files generated by the IS/1 Workbench for the VAX are fully compatible with files generated by VMS programs.

The command language of the IS/1 Workbench for the VAX utilizes an extended version of the UNIX Shell (command language interpreter). It contains extensions designed for use within Shell procedures (command files) that improve its usefulness to large programming groups, and make it more convenient for use as a high-level programming language. By utilizing the Shell as a programming language, Workbench users can eliminate a great deal of the programming tedium that often accompanies a large project. Many manual procedures can be quickly and conveniently automated. Because it is easy to create and use Shell procedures, each project that uses the IS/1 Workbench for the VAX can customize the general Workbench environment into one tailored to its own requirements, organizational structure and terminology.

Features of the Shell include:

- Ability to use any program as a command and to supply it with arbitrary character string arguments. File name arguments can be obtained from a pattern matching operation on the names of files in specified directories.

- Ability to execute from the Shell a program that may be either an image to be executed, a Shell command procedure or a DCL command procedure.

- Redirection of standard input and output permitting any program to run with file, terminal or other device input/output.

- Sequential execution of commands

- Parallel execution of commands with the output of one command connected to the input of another. This command chaining, called "pipelining", permits the construction of complex operations from sequences of IS/1 programs.

- Ability to run commands in "background" mode.

- Conditional execution: if, then, else and while constructions.

- String variables, and string and integer operations on those variables.

In the IS/1 Workbench for the VAX, a word/text processing system is provided that includes an editing system, a text formatting system, and spelling and typographical error detection facilities. The document preparation and text processing facilities of the IS/1 Workbench for the VAX include commands that automatically control pagination, style of paragraphs, line justification, hyphenation, multi-column pages, footnote placement, generation of marginal revision bars, and generation of tables of contents. Documents produced can include letters, memoranda, legal briefs, or specialized documents such as program run books. There are also facilities for formatting complex tables and equations.

IS/1 is a tremendously powerful software manipulation environment easily and rapidly combining primitive operations and commands. Rapid software prototyping and quick trial and error software experimentation evolve as a natural outcome of this ability to combine the various programs and commands using pipelines and filters.

Figure 3.9 provides an overview of IS/1 commands and application programs.

COMMAND	DESCRIPTION
abort	abort the typing of a file
admin	administer SCCS files
ar	archive and library maintainer
banner	print in block letters
bdiff	big diff
bfs	big file scanner
btt	convert a binary file to text
cal	print calendar
cat	concatenate and print
cb	C beautifier
cc	C compiler
cd	change working directory
chghist	change the history entry of an SCCS delta
chmod	change mode
chown	change a files owner
cmp	compare two files
code	print characters with their octal equivalents
col	filter reverse line feeds
comb	combine SCCS deltas
comm	print lines common to two files
copy	copy files and/or directories
cp	copy
cref	make cross reference listing
crypt	encode/decode
date	print and set the date
dcl	execute a DCL command from the Shell
del	delete files
delta	make an SCCS delta
deroff	remove nroff, troff, and eqn constructs
df	disk free
di	list contents of directory (verbose)
dict	check spelling of words in a file
diff	differential file comparator
diffmark	mark changes between versions of a file
du	summarize disk usage
ee	INed, the INTERACTIVE CRT text editor
ec	encrypt/decrypt files
echo	echo arguments
ed	text editor
edrestore	recover an ed session
eqn	typeset mathematics
equals	Shell assignment command
exit	terminate Shell command file
fd2	redirect file descriptor 2 (diagnostic output)

Figure 3.9 IS/1 Commands and Programs
(page 1 of 3)

COMMAND	DESCRIPTION
ffill	fill arbitrarily indented paragraphs of text
find	find files
fjust	fill & justify arbitrarily indented paragraphs of text
foreign.com	install a foreign command
gath	gather real and virtual files
get	get generation from SCCS file
goto	command transfer within a Shell procedure
grep	search a file for a pattern
help	ask for help
if	conditional command
INed	INTERACTIVE CRT text editor
kill	terminate a process
killall	kill all nonancestral processes
l	list with pagination
lex	generate programs for simple lexical tasks
ls	list contents of directory
m4	macro processor
make	make a program
man	print on-line documentation
mc	multicolumnar filter
mkdir	make a directory
msg	read messages in
mv	move or rename a file
neqn	typeset mathematics on terminal
newbin	rehash dirs in execution search sequence variable
news	print news items
next	new standard input
np	print the next page of file
nroff	text formatter
od	octal dump
onintr	handle interrupts in Shell files
pack	compress files
pcat	expand and concatenate compressed files
pr	print files
proofit	make a proof copy of documents
prt	print SCCS file
ps	process status
ptx	permuted index
pump	Shell data transfer command
pushes	print readable version of
pwd	working directory name
qtype	quickly type a copy of document
ratfor	rational FORTRAN dialect
recover	retrieve lost

Figure 3.9 IS/1 Commands and Programs
(page 2 of 3)

COMMAND	DESCRIPTION
reform	reformat text file
regcmp	regular expression compile
rjestat	*RJE* status and inquiries
rm	remove (unlink) files
rmdel	remove a delta from an SCCS file
rmdir	remove directory
rpl	replace all occurrences of a string in a file
sccsdiff	compare two versions of an SCCS file
sed	stream editor
send	submit RJE job
set	set a Shell variable
sh	Shell (command interpreter)
shift	adjust arguments in Shell command file
sleep	suspend execution for a interval
sno	SNOBOL interpreter
sort	sort or merge files
space	space and/or indent filter
split	split a file into pieces
stty	set terminal options
sum	print checksum of a file
switch	multi-way branch in Shell command file
tab	change blanks into tabs
tail	deliver the last part of a file
tbl	format tables for nroff or troff
tee	pipe fitting
time	time a command
to	send a message in
tr	transliterate
troff	format text for phototypesetting
ttab	convert a text file to binary
type	format and type a document
typo	find possible typographical errors
ufilter	filter underlines for terminal or line printer
uniq	report repeated lines in a file
unpack	expand compressed files
untab	change tabs into spaces
wait	await completion of process
wc	word count
what	identify files
whatsnew	compare file modification dates
while	iteration in a Shell command file
who	identifies users currently on system
write	write to another user
yacc	yet another compiler-compiler

Figure 3.9 IS/1 Commands and Programs
(page 3 of 3)

3.2.1.4.1 INed. INed is IS/1's interactive full screen editor for use with Interactive's INtext II terminals. INed provides a viewing window on the CRT screen in which all corrections, insertions, deletions, and other text editing functions are performed. Up to 10 viewing windows can be created on the screen. The files in the windows may be the same file (for viewing different parts of file while editing) or they may be different files. Editing functions can span windows and files, simplifying sectioning and concatenation of text between files. The INed screen editor is the user's most powerful tool for both document preparation and changes. While INed is capable of working with any type of ASCII RS232 terminal, use of the INtext II terminal pays great dividends in terms of user function and greatly reduces CPU overhead.

Arbitrarily defined rectangular regions of text may be moved, deleted, or duplicated. The user may scroll through a file, a page or any number of lines at a time, up or down. Scrolling left or right is also permitted. INed can handle a document as long as 720 pages of text, double spaced. Longer documents can be handled by creating multiple files.

All of INed's text editing features are available through the use of function keys (labeled on the INtext terminal). This feature speeds up the editing process by eliminating the need to enter verbose line-oriented commands. The use of function keys also reduces the length of time necessary for training. Most functions are invoked by pressing a single function key.

INed provides several levels of backup that prevent the inadvertent destruction of text files. For example, in the event of a hardware malfunction, the entire editing session can be reproduced, often without loss of a single keystroke. All lines deleted during an editing session are saved for potential recall until the end of that session. In addition, an original copy of every file edited is automatically saved under a backup file name.

Many system utilities can be executed directly from INed. They can be invoked to process a line, a paragraph, or entire file. The results replace the processed text in the file and on the CRT so that the user can immediately view the calculation or transformation. In this manner, paragraphs can be filled or justified, columns of numbers totalled, or user written programs accessed to perform specialized functions on the text being displayed.

3.2.1.4.2 INword. A need exists to ease the burden of software document generation that is necessary to support the various software projects within DMA. This capability is

realized by using the word processing system (INword) which works closely with the INed screen editor to allow rapid formatting of documents with a minimum of CPU time. INword also works quite well for producing memos, letters, or papers. To illustrate the use of INword, a typical document editing session follows:

1. The text of the document is entered using INed.

2. The text formatting commands are placed with the text to provide page control, centering, and justification.

3. The "proofit" command produces a "proof" file that can be windowed with INed to show the appearance of the document on the printed page.

4. Using an alternate display window on the original document, corrections are made and step 3 is repeated.

5. The document is printed in final form on a hardcopy printer.

The INed editor itself has a number of built-in page formatting functions and simple memos can be formatted using only INed. Figures 3.10 and 3.11 highlight the features and added utilities of the word processor, INword.

- justification of either or both margins
- automatic hyphenation using a sophisticated logic program
- suppression of automatic hyphenation
- footnotes which can carry over to the next page if excessively long
- automatic page numbering at top or bottom of page
- indents, permanent and temporary
- underlining
- conditional insertion and deletion of text
- automatically numbered and positioned footnotes
- automatically numbered headings
- even-odd page differentiation capabilities
- centering
- specification of multiline headers
- specification of multiline footers
- keeping a block of text or a table on a single page
- specification of page length (if default length not desired)
- specification of left and right margins (if default margins not desired)
- forcing the beginning of a new page
- extension of a particular page by a few lines
- automatic widow and orphan control
- single or double spacing
- specification of top and/or bottom margins including multiline headers and footers (if default header or footer not desired)

Figure 3.10 INword Features

dict: identifies possible typographical and spelling errors by comparing all the words in the copy with those stored in a 50,000 word dictionary. Up to 80,000 additional words may be added to the dictionary.

diff: lists the differences between two files. This will explicitly point out all of the changes made.

comm: finds the lines in common between two files. This will aid in pointing out repetitive information.

grep: searches through a file or files for a particular string or pattern. This will help locate those sections that need revision.

proofit: creates a complete formatted version of the file exactly as it will look when it is run on a printer. The original file can be accessed while viewing the proof file as either an alternate file or a second window on the screen.

Figure 3.11 INword Utilities

3.2.1.4.3 SCCS. The Source Code Control System (SCCS) is an integrated set of commands designed to help software development projects control changes to source code and to files of text (e.g., manuals). It is equally useful for tracking and controlling changes to other documents that are subject to frequent revision. It provides facilities for storing, updating and retrieving, by version number or date, all versions of source code modules or of documents, and for recording who made each change, when it was made and why. SCCS is designed to solve many of the source code and documentation control problems that software development projects encounter when customer support, system testing and development are all proceeding simultaneously.

Some of the main characteristics of SCCS are:

- The exact source code or text, as it existed at any stage of development or maintenance, can be recreated at any later time.
- All releases and versions of a source code module or document are stored together, so that common code or text is stored only once.
- Releases in production or system-test status can be protected from unauthorized changes.
- Enough identifying information can be automatically inserted into source code modules to enable a user to identify the exact version and release of any such module given only the corresponding load module or its memory dump.

3.2.1.5 FAVS(RXVP80). The Fortran Automated Verification System (FAVS) is a tool for analyzing source programs written in FORTRAN or DMATRAN(IFTRAN). FAVS is currently in use at DMA centers on UNIVAC equipment. RXVP80 is the commercial version of FAVS and is available on the VAX. Though there are differences between the two tools, the basic capabilities, as outlined below, are provided by both tools. These tools aid in improving the quality and reliability of software by providing:

- Indented listings of source programs.
- Static analysis to detect inconsistencies in program structure, in the use of variables, and in calling parameters.
- Automated documentation.
- Instrumentation of source code.
- Analysis of testing coverage.
- Retesting guidance.

With the use of FAVS or RXVP80, assistance is provided to the user from the very early stages of implementation, through system integration, testing, documentation, and maintenance. Figure 3.12 illustrates the steps in validating a program with FAVS or RXVP80.

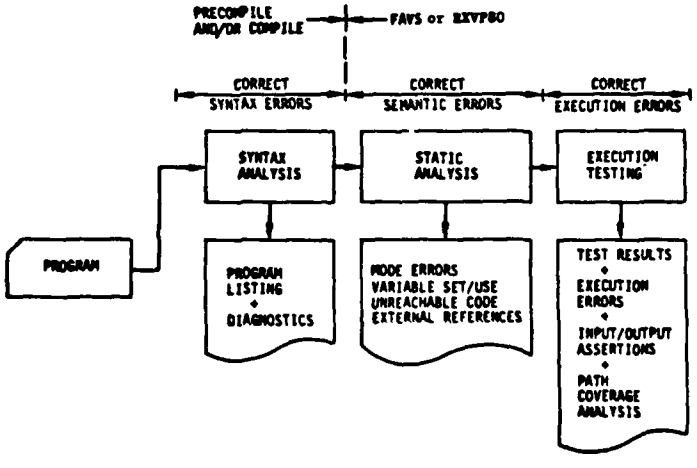


Figure 3.12 Steps in Validating a Program with FAVS or RXVP80

Figure 3.13 shows how FAVS or RXVP80 fits into the software development cycle to augment software analysis and testing. The additional features are indicated by diagonal lines. The user's source code can be analyzed and the results will be presented in reports which help the user decide if acceptance criteria are being met. The tools can also instrument source code prior to test execution and provide a test coverage analysis of the behavior of the program during execution.

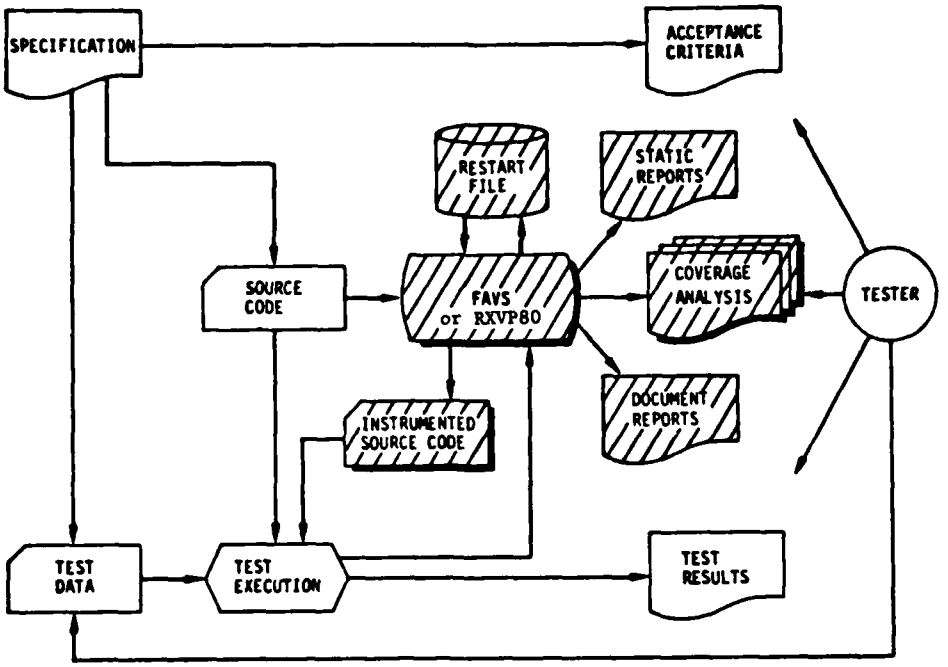


Figure 3.13 Software Analysis and Testing Augmented by FAVS or RXVP80

3.2.1.6 CAVS. The COBOL Automated Verification System (CAVS) is a tool for analyzing source programs written in COBOL. CAVS is an aid in improving the quality and reliability of software by providing as a minimum the following set of capabilities.

- o Program analysis and error detection
- o Documentation production
- o Incremental updating of the Project Library
- o Instrumentation
- o Execution analysis
- o Coverage assistance
- o Selective application of these functions

CAVS is intended to improve the reliability of COBOL programs by working as an aid during the coding, testing, implementation and maintenance phases of a project. Figure 3.14 illustrates the role of CAVS in developing systems.

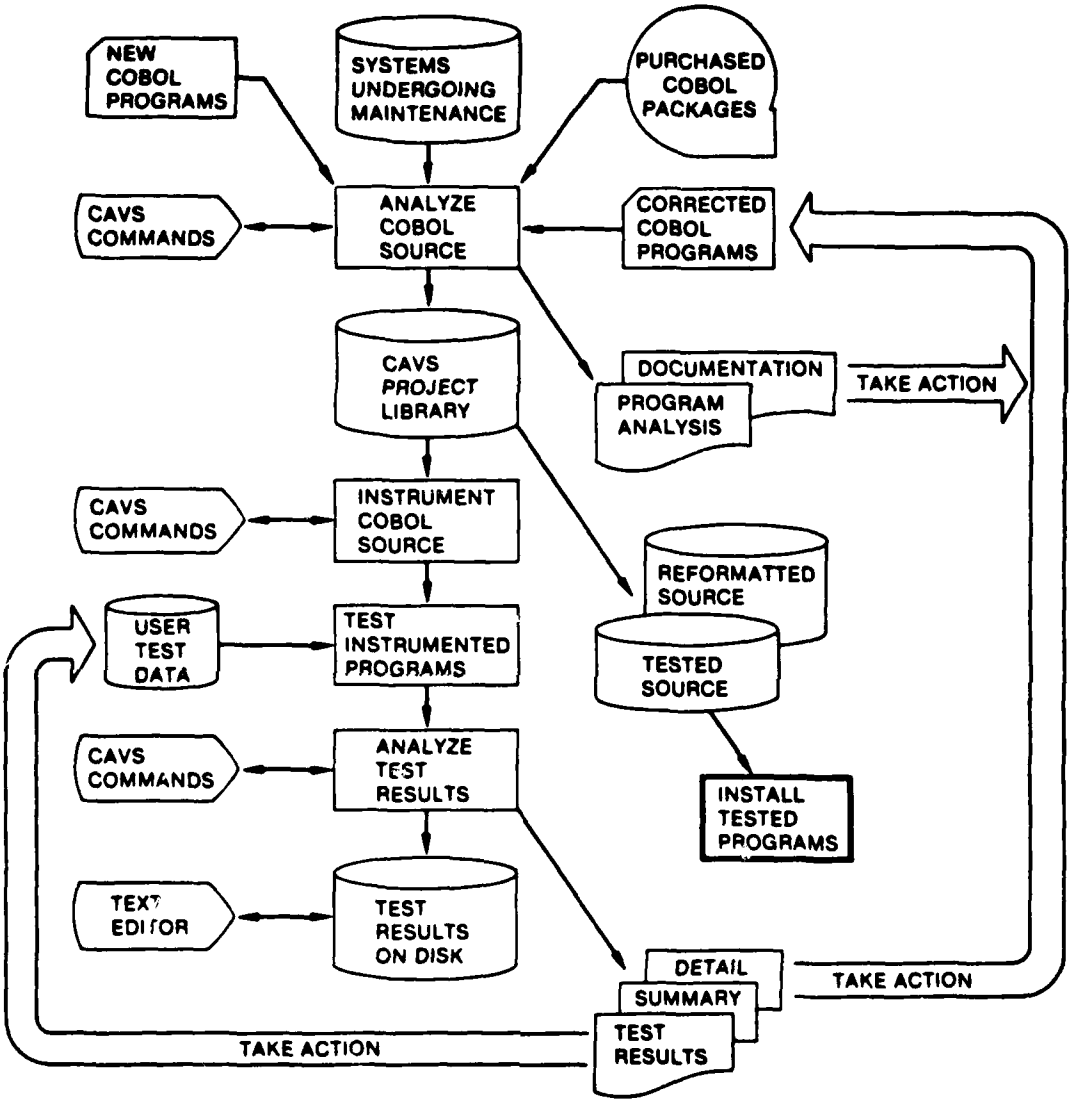


Figure 3.14 CAVS Use in Developing Systems

3.2.1.7 VUE. The VUE tool is an integrated, menu-driven system for project planning and control. It is based on networking techniques. VUE performs time analysis, cost analysis, resource analysis, resource allocation, report processing, including network plots, and maintenance/updating of VUE mass storage files. Outputs include the following reports:

o CALENDAR SPECIFICATIONS - Shows the project start date, which days are in the standard work week, and lists holidays and special work days (up to 500 of each). Calendar span is five years.

o NODE SCHEDULE - When VUE analyses an I-J (activity on arrow) network, this report shows early and late occurrence times for the nodes between activities.

o PREDECESSOR REPORT - Reports all the predecessors of the selected activities and the lag times specified (for precedence networks).

o START AND FINISH NODES REPORT - VUE allows multiple start and finish nodes, and these are listed in this report. Also listed are any nodes which may be unconnected to the network.

o RESOURCE REPORT - Resource levels for up to ten resources may be specified for each activity. This report gives a daily report of resource levels needed to keep the project on schedule.

o ACTIVITY TIMES REPORT - Compares an activity's scheduled times with its target start, target finish, and completion times.

o COST ACCOUNTING REPORT - Each activity can carry up to four separate costs, and they can be positive or negative for credit/debit accounting. This report gives monthly subtotals and a running total for the activities reported.

o CURRENT WORK REPORT - Lists those activities which should be worked on during the current time window (e.g. this week, this month).

o PROGRESS REPORT - Reports on time and behind schedule activities.

o BAR CHART REPORT - Shows schedule for activities in a graphical form. Critical activities are clearly indicated. Bar chart scale is variable, and can either be specified by the user or left to automatic program control.

o SCHEDULE REPORT - Shows early start, early finish, late start, late finish, and three floats for each activity.

3.2.3 General Support.

3.2.3.1 HYPERGRAPHICS. The HYPERGRAPHICS tool is a microcomputer based system for the creation and delivery of lecture material. Applications include formal presentations and classroom training. An additional capability is the preparation and use of interactive lesson plans used on an individual basis for more specialized self-paced training.

Sample features of the system are as follows:

- o Screen mode editor
- o Box, line and figure composition facilities
- o Color graphics embedding facility
- o On-line help
- o Dynamic delivery functions
- o Highlighting function for presentation emphasis
- o Permanent inverse video for highlighting

HYPERGRAPHICS is available on an APPLE II Pascal based microcomputer. Floppy disks are used for the storage of lecture material, (approximately 145 pages per diskette). In classroom lectures a large screen TV may be used for display though a smaller screen TV would be more practical for the preparation of a lecture. The lecture material consists of page frames displayed in predetermined or instructor/presenter directed order. Each page in turn consists of 20 lines of 40 characters each. Upper and lower case character are available along with most common special characters.

HYPERGRAPHICS operates in two modes. First the editing functions are used in the creation and/or modification of lecture material. The second set of functions are used in the presentation of the material. Figure 3.15 lists the twelve commands comprising the command facility.

COMMAND	ACTION
N	Transfer to the next page
Z	Transfer to page zero
P	Transfer to the previous page
J	Transfer to user selected page
0-9	Transfer to numbered reference
ESC 1-5	Execute Pascal program reference
G	Cycle next page sequence until key press
R	Reverse screen color
M	Activate line marker
H	Highlight marked line
;	Move marker up one line
/	Move marker down one line
C	Change system parameters
L	List page on printer
O	List several pages
?	Display commands
E	Edit current page
Q	Exit program

Figure 3.15 HYPERGRAPHICS Command List

The first five commands are used most often during delivery of a lecture. N causes the display of the page linked as the next logical (not necessarily physical) page. The zero page is displayed when the system is initialized and usually serves as a table of contents for the material in the remainder of the file. The Z command is used to return to page zero from any other page. This command is particularly useful in review situations where rapid access to different pages is needed. As different pages are accessed, the program keeps track of the traversal order. The P command is used to review these previously displayed pages. The J command is used to select a page by number.

Pages can have up to ten references to other pages. These references are usually shown on the screen as a single digit surrounded by a single lined box. Pressing one of the numeric keys corresponding to a reference causes the selected page to be presented.

Up to five Pascal program references can be made from each page. These references are usually shown on the screen as a single digit surrounded by a double lined box. HYPERGRAPHICS chains to these programs sending information needed to reenter the program and displaying the current page. If the referenced program chains back to HYPERGRAPHICS, this information is used to select the proper file and page.

Normally the images are displayed in black on a white background. Some pages are easier to read with the display colors reversed. This is accomplished by the R command. A marker can be used to call attention to a particular line on the screen. The line being marked can be highlighted by reversing its colors.

If the system has a printer attached, hardcopy of the page being displayed can be obtained with the L command. The O command is used to list a sequence of pages.

The HYPERGRAPHICS system is very easy to learn. A short session is all that is needed to learn how to traverse pages and use the system. During this training time, the ? command can be used to display a complete list of the commands and their functions. A prompt line that lists the possible commands can be displayed at the bottom of the screen. This and other system parameters can be changed by the C command. Normally the prompt line is not displayed because it might be confused with the page material.

The final command, Q, is used to terminate execution of the program.

The normal traversal functions are used to move within the network of pages while material is being created or edited. The edit mode is entered by the E command. The editor has a full array of screen editing functions. The editor functions allow the user to insert a blank line moving the rest of the page down, to delete a line. The editor also allows the user to alter lines by inserting, deleting or replacing characters. Characters can be entered in either black on white or white on black modes. These modes can be mixed in a line. Full cursor control can be used to move to any location on the screen while using this edit function.

Other edit functions provide a means of changing the next page and the other references for the page being edited and to load a copy of another page from the disk. This feature is particularly useful when a sequence of pages is being created with relatively minor changes from one page to the next.

After a page has been edited, the changes can be written to disk or disregarded. Thus changes made during a lecture can be treated as temporary or permanent.

3.3 Interfaces. Communication links must be established using standard user interface protocols or I/O channels. Since multiple VAX-11/780's (3) are recommended for each center, a standard BISYNCH protocol should be used if possible. The MUX/200 VAX system is recommended for communications between the VAX development systems and the UNIVAC production mainframe. The DECnet LAN is recommended for communications amongst the VAX systems within each DMA center. Descriptions of MUX 200 and DECnet can be found in Figure 3.2.

3.4 Security and Privacy. This section does not apply to this specification.

3.5 Controls. This section does not apply to this specification.

SECTION 4. DESIGN DETAILS

This section does not apply to this specification.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.